sklearn.cross_validation.StratifiedKFold

» *class* sklearn.cross_validation.**StratifiedKFold**(*y*, *n_folds=3*, *shuffle=False*, *random_state=None*) [source]

Stratified K-Folds cross validation iterator

Provides train/test indices to split data in train test sets.

This cross-validation object is a variation of KFold that returns stratified folds. The folds are made by preserving the percentage of samples for each class.

Read more in the User Guide.

Parameters:

y : array-like, [n_samples]

Samples to split in K folds.

n_folds : int, default=3

Number of folds. Must be at least 2.

shuffle : boolean, optional

Whether to shuffle each stratification of the data before splitting into batches.

random_state : None, int or RandomState

When shuffle=True, pseudo-random number generator state used for shuffling. If None, use default numpy RNG for shuffling.

See also:

`LabelKFold`

K-fold iterator variant with non-overlapping labels.

Notes

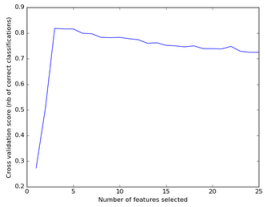
All the folds have size trunc(n_samples / n_folds), the last one has the complementary.

Examples

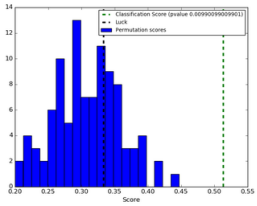
```
>>> from sklearn.cross_validation import StratifiedKFold
>>> X = np.array([[1, 2], [3, 4], [1, 2], [3, 4]])
>>> y = np.array([0, 0, 1, 1])
>>> skf = StratifiedKFold(y, n_folds=2)
>>> len(skf)
2
>>> print(skf)
sklearn.cross_validation.StratifiedKFold(labels=[0 0 1 1], n_folds=2,
                                         shuffle=False, random_state=None)
>>> for train_index, test_index in skf:
...     print("TRAIN:", train_index, "TEST:", test_index)
...     X_train, X_test = X[train_index], X[test_index]
...     y_train, y_test = y[train_index], y[test_index]
TRAIN: [1 3] TEST: [0 2]
TRAIN: [0 2] TEST: [1 3]
.. automethod:: __init__
```

»

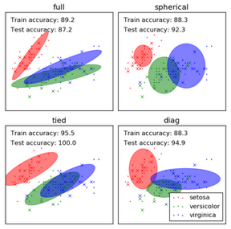
Examples using `sklearn.cross_validation.StratifiedKFold`



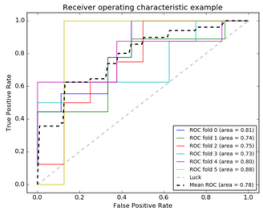
Recursive feature elimination with cross-validation



Test with permutations the significance of a classification score



GMM classification



Receiver Operating Characteristic (ROC) with cross validation